ยุคลอาจ ชาญพานิชกิจการ

Advance Clipper กับการสร้างเมนูที่สะดวกรวดเร็ว

ขั้นตอนที่จะทำให้คุณสามารถสร้างเมนูได้อย่างรวดเร็วและเปลี่ยนแปลงได้ง่ายมาก

สวัสดีครับเป็นยังไงกันบ้างจากเรื่อง Clipper ในบทความก่อนหน้านี้ คงจะได้ผ่านตากันไปบ้างแล้ว มีผู้อ่านโทร.มาถามถึงปัญหาบางจุดของโปรแกรม ซึ่งต้องขอมรับว่าเป็นความผิดของผมเองที่ไม่ได้ตรวจสอบอย่างละเอียด เรื่องมันมีอยู่ว่า คำสั่ง Setlastkey(0) นั้นไม่ใช่ Lib มาตรฐานของ Clipper แล้วผมก็ไม่ได้ฉุกคิดไปว่าคุณผู้อ่านทั้งหลายคงจะไม่มีไอ้เจ้าฟังก์ชันตัวนี้แน่เพราะมันเป็นฟังก์ชันส่วนตัวของผมเอง (สร้างเองไงครับ) แต่..... มันก็มีทางออกอยู่ คือถ้าคุณคอมพายล์โปรแกรมแล้วมันไม่ผ่านเพราะไอ้เจ้าคำสั่งนี้ละก็ .... ให้คุณทำการแก้ไขเป็นคำสั่ง KEYBOARD CHR(0) แทนนะครับรับรองผล 100 เปอร์เซ็นต์แน่นอน คราวนี้ผ่านแน่

โดยส่วนใหญ่สมัยนี้การเขียนโปรแกรมจะอิงไปบนวินโดวส์กันเป็นส่วนใหญ่ ส่วนโปรแกรมบนดอสนั้นคงจะน้อยลง แต่ยังไงก็ยังไม่ได้ล้มหายตายจากไปเสียทีเดียว เพราะระบบงานทางด้านคอมพิวเตอร์นั้น การจะโยกย้ายหรือเปลี่ยนแปลงมันจะต้องค่อยเป็นค่อยไป ฉะนั้นคุณยังสามารถที่จะเขียนโปรแกรมบนดอสได้ครับ ถึงอย่างไรก็ตาม ผมว่าคุณควรจะศึกษาการพัฒนาโปรแกรมบนวินโดวส์เอาไว้ด้วย เรียกว่าได้ทั้งสองอย่างแหละดีที่สุด โม้ซะยืดยาว เข้าเรื่องเลยดีกว่า

การพัฒนาโปรแกรมด้วย Clipper นั้น การสร้างเมนู เอาอย่างง่ายก่อนแล้วกัน คุณต้องใช้คำสั่ง

```
"@ nRow,nCol PROMPT <Display Text>"
```

อยากได้อะไรบ้างก็เขียนเรียงกันลงมาเรื่อย ๆ ถ้าต้องการให้เมนูมีการเรียกเมนูย่อยด้วย ก็ต้องเขียน Loop ซ้อนกันไปเรื่อย ๆ อย่างที่สองก็คือใช้ฟังก์ชัน ACHOICE เข้ามาช่วยแทนคำสั่งข้างบนแน่นอนว่ามันก็จะยังต้องมี Loop ซ้อนเข้าไปอีก และถ้าในกรณีที่เป็นระบบโปรแกรมใหญ่หน่อย มีหลายเมนูเหลือเกินคราวนี้แหละครับยุ่ง เราจะมาหาทางออกในเรื่องนี้กันครับ

การแก้ความยุ่งยากตรงนี้เราจะใช้ Database File ของ Clipper เป็นตัวเก็บรายละเอียดของเมนู ฟิลด์ที่จะต้องสร้างขึ้นมาเก็บข้อมูลต่าง ๆ สำหรับเอาไว้สร้างเมนูก็จะมีดังตารางที่ 1

จากตารางฟิลด์เมนู คุณสามารถที่จะเพิ่มเติมในส่วนที่ต้องการเข้าไปได้อีกโดยคุณอาจจะเพิ่มในส่วนของ Password และส่วนของผู้ที่มีสิทธิเข้าสู่เมนูได้ โดยเฉพาะในส่วนของ Password คุณควรจะเก็บโดยการเข้ารหัสเอาไว้ด้วย คงจะทำกันได้นะครับ ถ้ามีโอกาสจะเขียนในส่วนของวิธีการเข้ารหัสโดยใช้ Clipper

หลักจากที่คุณได้ทำการสร้างไฟล์ที่มีฟิลด์ตามตารางนี้แล้ว ต่อไปคุณจะต้องเขียนโปรแกรมขึ้นมาอ่านมันแล้วทำให้เป็นเมนูให้ได้ มันก็ไม่ยากเท่าไรหรอกครับ โปรแกรมตัวนี้ผมได้กำหนดวิธีการที่ใช้จัดการเมนู 2 วิธี วิธีแรกก็คือใช้ฟังก์ชัน ACHOICE เป็นตัวจัดการเมนู และวิธีที่สองใช้ ฟังก์ชันที่ผมเขียนขึ้นมาเอง ที่ต้องเขียนขึ้นมาเพิ่มเพราะว่าเวลาใช้ฟังก์ชัน ACHOICE ถ้าเรามี Sub Menu แล้ว กรณีที่ยกเลิก Sub menu นั้น รายการที่เลือกไว้จะมาอยู่ ณ ตำแหน่งแรกของบรรทัดที่กำหนด ไม่รู้ว่าจะเข้าใจหรือเปล่า ยังไงก็ลองดูแล้วกันนะครับ สรุปก็ไม่

ตารางที่ 1

Field	Type	Width	Decimal	Description
MENUNO	N	8	4	หมายเลขเมนู
MENULABEL	C	60		ชื่อเมนู
MENUFLAG	C	1		คุณสมบัติของเมนู
MENUCOLOR	C	30		สีของเมนู
MENUCALL	C	30		ชื่อที่ต้องการเรียก
MENUDIR	C	30		สถานที่เก็บอ่านโปรแกรม
MENUPOS	C	15		ตำแหน่งของเมนู

รู้จะแก้ไขยังไงได้เพราะถ้าทำไปแล้วดูมันไม่ค่อยจะดีเท่าไร เพราะฉะนั้นเขียนเองดีกว่า และนี่คือที่มาของส่วนตัวจัดการเมนู ตัวโปรแกรมจะประกอบไปด้วยสองส่วนใหญ่ คือ ตัวอ่านค่าเมนูและกำหนด และตัวจัดการเมนู

วิธีการทำงาน ถ้าคุณใช้งานในลักษณะที่ใช้คำสั่ง DO เพื่อเรียกโปรแกรมอื่น ต้องทำการเปลี่ยนแปลงไปใช้ Script.LNK ไฟล์แทน วิธีการสร้าง Script.LNK นั้น มีรูปแบบการสร้างดังนี้

ตัวอย่าง SCRIPT.LNK ไฟล์

```
1. FILE MYMAIN  <-- ชื่อ Main โปรแกรม
2. OUT  MYMAIN  <-- ชื่อ EXE File ที่ต้องการ
3. BEGINAREA
      4. FILE PROGRAM1
      5. FILE PROGRAM2  <-- ชื่อโปรแกรมย่อย .PRG ตัวอื่น
      6. FILE PROGRAM3
7. ENDAREA
8. LIB CLIPPER,LIBNAME2,  <-- ชื่อ Library File ที่นอกเหนือไปจาก
   LIBNAME3,LIBNAME4        Library มาตรฐานของ Clipper
```

ในบรรทัดแรกเป็นชื่อโปรแกรมหลักซึ่งก็คือเมนูที่เราเขียนขึ้นมานี่แหละครับ คุณต้องไปเปลี่ยนชื่อเอาเองตามชอบนะครับ บรรทัดที่สองเป็นชื่อ EXE File คือหลังจากทำการ Link แล้วต้องการจะให้ได้ชื่ออะไรก็มาใส่ที่นี่บรรทัดที่สามและเจ็ดเป็นข้อบังคับที่จะต้องใส่ตามนั้น ส่วนบรรทัดที่แปดเป็นชื่อ Library ที่คุณต้องการเรียกใช้นอกเหนือไปจาก Library มาตรฐานของ Clipper เอง วิธีการใช้งาน Script.LNK นี้จะใช้หลักจากที่คุณ Compile เสร็จแล้ว คำสั่งการใช้จะมีรูปแบบดังนี้

```
RTLINK @Script.LNK
```

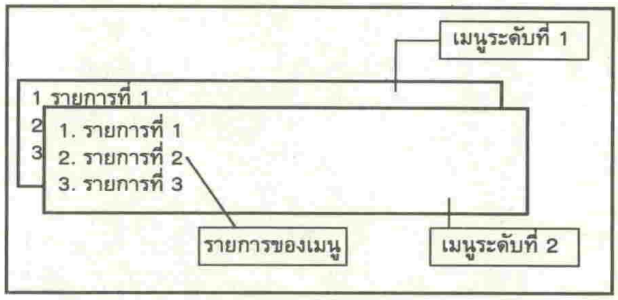
หรือถ้าใช้ Blinker

```
BLINKER @Script.LNK
```

อธิบายโปรแกรมและหลักการ

ก่อนอื่นต้องมาทำความเข้าใจวิธีลำดับเมนูเราจะแบ่งตัวลำดับซึ่งก็คือฟิลด์ MENU_NO นั่นเอง แบ่งออกเป็น 3 ส่วนจะมีรูปแบบดังนี้

งงไหมครับผมจะอธิบายให้ชัดเจนขึ้น รายการแรกในเมนูระดับที่หนึ่งรายการที่หนึ่งเราจะกำหนดให้เป็น 001.0001 ระดับคือชั้นของเมนูนะครับและรายการคือตัวเลือกของเมนู ดังรูปที่ 1



รูปที่ 1

ถ้าเรามีรายการต่อไปในเมนูระดับที่ 1 ก็จะเป็น 1.0002,1.0003 ฯ เรียงตามกันไปเรื่อยนะครับ ในเหตุผลเดียวกันถ้าคุณมีเมนูในระดับที่ 2 รายการที่หนึ่งก็จะเป็น 2.xx01,2.xx02,2.xx03 ที่สองตัวกลางเป็น xx นั้นเพราะว่าตัวเลขนี้จะขึ้นกับรายการของเมนูในระดับที่หนึ่ง สมมติว่าเมนูในระดับที่ 2 ของคุณจะถูกเรียกใช้โดยรายการที่ 2 ของเมนูระดับที่ 1 เราก็จะกำหนดได้เป็น 2.0201 คือเป็นเมนูในระดับที่ 2 ถูกเรียกจากเมนูระดับที่ 1 รายการที่ 2 และเป็นรายการที่ 1 ของเมนูในระดับที่ 2 คราวนี้เข้าใจหรือยังครับ เพราะฉะนั้นเราจึงสามารถที่จะแยกเมนูออกเป็นส่วน ๆ ได้ โดยการแบ่งแยกเช่นนี้ สรุปเราสามารถที่จะมีจำนวนเมนูสูงสุดได้เท่ากับ 99 ระดับรายการต่อเมนู และแต่ละรายการก็จะมี 99 รายการอีกเช่นกัน

การออกแบบเมนูในลักษณะนี้จะต้องมีโปรแกรมที่ทำหน้าที่สร้างเมนูให้เรา ซึ่งผมก็ได้ให้ตัวโปรแกรมที่ทำหน้าที่สร้าง คือเราสามารถที่จะกำหนดรายละเอียดได้ง่ายขึ้น หรือถ้าคุณพอใจจะใช้ DBU ในการสร้างก็ได้นะครับ วิธีใช้ให้คุณพิมพ์โปรแกรม Menu.prg ตามตัวอย่างนี้และทำการ Compile ให้เป็น EXE ไฟล์ ก่อนจะเรียกใช้โปรแกรมตัวนี้ให้สร้าง Database ไฟล์ขึ้นมาก่อนนะครับ ตามตัวอย่างตารางที่ 1 เพราะโปรแกรม Menu.prg จะมีการเรียกใช้ Database ไฟล์นี้

โปรแกรม Menu.prg นี้จะแบ่งการทำงานออกเป็นสองส่วน คือ การแก้ไขเมนู (Edit) และการทดสอบเรียกใช้เมนู (Run) เพื่อความเข้าใจในการสร้างเมนูผมอยากให้คุณเลือก Edit และพิมพ์ข้อมูลเข้าไปตามตารางที่ 2

ตารางที่ 2

MENUNO	MENULABEL	MENUFLAG	MENUCOLOR	MENUCALL	MENUDIR	MENUPOS
1.0001	File	M	N/W,W/N,,,N/W	2.0101		07,06,17,60
1.0002	Edit	M	2.0201			
1.0003	View	P	TEST			
1.0004	Insert					
2.0101	New	M	N/BG,N/W,,,N/BG	3.0201		09,10,19,65
2.0102	Open					
2.0201	Undo Clear	M	GR+/B,N/W,,,GR+/B	3.0101		09,10,19,65
2.0202	Repeat Clear					
3.0101	Normal		N/G,N/W,,,N/G			11,12,21,68
3.0201	OutLine		GR+/B,N/W,,,GR+/B			11,12,21,68

เป็นตัวบอกลักษณะของเมนู ว่าเป็นถ้า Enter ที่เมนูนี้แล้วจะเป็นการเรียกเมนูย่อยหรือ เรียกโปรแกรมอื่นๆ ขึ้นมาทำงาน ถ้าเป็น M คือ เรียกเมนูย่อย ถ้าเป็น P คือ เรียกโปรแกรม

กรณีที่ Flag เป็น M ค่า Call เราจะต้องใส่เป็นหมายเลขลำดับต่อไปของเมนูที่ต้องการให้เรียกใช้ วิธีกำหนด หมายเลขให้ทำตามที่ได้อธิบายเอาไว้ก่อนหน้านี้ ถ้าไม่เข้าใจลอง Run โปรแกรมดูแล้วจะเข้าใจนะครับ กรณีที่เป็นการเรียกโปรแกรม ค่าตรงนี้ก็สามารถใส่ชื่อโปรแกรมได้เลย

หลังจากที่พิมพ์ข้อมูลเข้าไปตามนี้แล้ว ให้เลือกเมนู Run แล้วศึกษาดูนะครับ

โปรแกรมตัวต่อไปเป็นตัวอ่านเมนู ชื่อโปรแกรมคือ RDMENU.prg ซึ่งก็คือตัวโปรแกรมที่ทำหน้าที่ Run นั่นเอง เพียงแต่เราเอามารวมไว้ในส่วนของโปรแกรมสร้างเมนูเพื่อความสะดวกในการสร้างและทดลองใช้ ให้คุณพิมพ์โปรแกรมตัวนี้ลงไปโปรแกรมตัวนี้จะทำการแปลงข้อมูลที่คุณพิมพ์ในตารางที่ 2 มาเป็นเมนูให้เราใช้งานกัน โดยการทำงานนั้นเราจะใช้ Array เพียงชุดเดียวในการทำงาน โดยการ ReLoad ข้อมูลตามที่ถูกใช้งานในแต่ละเมนู เพื่อความประหยัดในการใช้งานหน่วยความจำ และจะทำให้เรามีเมนูหลักและมีเมนูย่อยได้อีกมาก ถ้าค่าของ 99 ไม่พอ ก็เพียงแต่คุณขยายค่าให้เป็น 3 หลักก็จะได้ 999 เมนูแล้ว โดยที่ตัวอ่านเมนูสามารถที่จะรับค่านั้นได้โดยไม่ต้องลุ้นว่าหน่วยความจำจะพอหรือไม่ ตัวโปรแกรมจะมีการเก็บค่าบางอย่างก่อนทำการ Load เมนูใหม่ขึ้นมาเพื่อที่ว่ามีการเรียกกลับแล้วจะสามารถอ่านข้อมูลขึ้นมาได้ถูกต้อง

โปรแกรมที่ทำรูปแบบเมนู เป็นโปรแกรมที่เราจะ Include เข้ามาอยู่ในไฟล์ชื่อ c_menu.ch โดยโปรแกรมตัวนี้จะทำหน้าที่กำหนดรูปแบบเมนูให้เรา ก็คือ ฟังก์ชัน cm_menu

วิธีการใช้งาน

ให้คุณทำการเปลี่ยนชื่อ RDMENU.prg เป็นชื่อโปรแกรม main ในระบบของคุณ โดยวิธีการ copy ดูที่ตัวอย่างต่อไปนี้ ในส่วนของชื่อโปรแกรมย่อยตัวอื่น ๆ นั้นจะเป็นชื่อโปรแกรมที่คุณเขียนขึ้นมาเอง ชื่อโปรแกรมนี้คุณจะต้องนำไปใส่ไว้ใน MenuCall ของข้อมูลเมนูที่คุณกำหนดขึ้นมา หลังจากที่คุณทำการสร้าง Script ตามที่ต้องการได้แล้ว ให้คุณทำการคอมพายล์ แน่นอนคุณจะต้องคอมพายล์โปรแกรมทุกตัว เพื่อที่จะเอาไว้ Link วิธีการ Link ให้ใช้ Rtlink @<File Script.lnk> เช่นถ้าผมมี Script.lnk ที่ชื่อ Mainpro.lnk ผมก็ต้องใช้คำสั่ง Link ดังนี้ Rtlink @Menupro.lnk แล้วคุณก็จะได้ Exe ไฟล์ที่ทำงานได้ออกมาตามชื่อที่คุณกำหนดไว้ที่ OUT

อาจจะมีหลายขั้นตอนสักนิดแต่ผมเชื่อว่าคงจะไม่ยากเกินไปนัก เพราะการกำหนดอย่างนี้จะทำให้คุณสามารถที่จะสร้างเมนูได้อย่างรวดเร็วและเปลี่ยนแปลงได้ง่ายมาก ติดปัญหาอย่างไรก็ถามมาได้นะครับ ยินดีที่จะตอบให้ ฉบับนี้แค่นี้ครับ

```
1. FILE MYMAIN <---------- โปรแกรม RDMENU.prg ที่เปลี่ยนชื่อแล้ว
2. OUT  MYMAIN <---------- ชื่อ EXE File ที่ต้องการ
3. BEGINAREA
   4. FILE PROGRAM1
   5. FILE PROGRAM2<---------- ชื่อโปรแกรมย่อย .PRG ตัวอื่น
   6. FILE PROGRAM3
7. ENDAREA
8. LIB CLIPPER,LIBNAME2 <- ชื่อ Library File ที่นอกเหนือไปจาก
   LIBNAME3,LIBNAME4       Library มาตรฐานของ Clipper
```

ลิสติงที่ 1 โปรแกรมสร้างเมนูและทดสอบเมนู (MENU.PRG)

```
Program Menu.Prg
*/
LOCAL NCHOICE,ALFILE,FLIST,SCR
CLEAR

/* DEFINE KEY */

#define K_HOME              1   //   Home, Ctrl-A
#define K_ENTER            13   //   Enter, Ctrl-M
#define K_ESC              27   //   Esc, Ctrl-[
#define K_INS              22   //   Ins, Ctrl-V
#define K_DEL               7   //   Del, Ctrl-G
#define K_LEFT             19   //   Left arrow, Ctrl-S
#define K_RIGHT             4   //   Right arrow, Ctrl-D

SET delete on
set defa to

       DBUSEAREA(,,"MENU")
       DBSETINDEX("MENU")
       SETCOLOR("GR+/B")
       DEVPOS(1,0)
       DEVOUT(REPL(" ",80))
       DEVPOS(2,0)
       DEVOUT(REPL(" ",80))
       DEVPOS(3,0)
       DEVOUT(REPL(" ",80))
       DISPBOX(0,0,4,79)
       DEVPOS(1,(80-LEN("Menu Create Version 1.0a"))/2)
       DEVOUT("Menu Create Version 1.0a","GR+/B")
       DEVPOS(3,(80-LEN("Programming By AONG"))/2)
       DEVOUT("Programming By AONG","GR+/B")
       dispbox(5,0,9,16)
     WHILE(.T.)
       NCHOICE := Achoice(6,1,8,15,{" Edit "," Run "," Quit "})

       DO CASE
          CASE NCHOICE == 1
               agBrowse(,,,,{"@zr 999.9999","@S30","@s30","@s30","@s30","@s30","@!"})
          CASE NCHOICE == 2
               SCR := SAVESCREEN(0,0,24,79)
               DO RDMENU
               RESTSCREEN(0,0,24,79,SCR)
               SETCOLOR("GR+/B,N/W,,,GR+/B")
               DBUSEAREA(,,"MENU")
               DBSETINDEX("MENU")

          CASE NCHOICE == 3
               EXIT
       ENDCASE
     ENDDO
     dbcloseall()
     SETCOLOR("W/N")
     CLEAR
RETURN


//
PROCEDURE agBrowse(aField,aHead,cfunc,cTil,aPic)
local  scr := savescreen(0,0,24,79),oldcolor:=setcolor(),oldorder:=indexord()
private cBorder := "",aPicture := aPic
       if cTil == NIL
          cTil = "Edit File : "+ALIAS()
       endif
       setcolor("N/BG,BG/N,,,N/BG")
       @ 0,0 CLEAR TO 24,79
       DISPBOX(0,0,2,79,,)
       devpos(0,3)
       devout(cTil)
       devpos(1,3)
       devout("[ Record no. # :              ]")
```

ลิสติงที่ 1 (ต่อ)

```
        DISPBOX(22,0,24,79,,)
        devpos(23,3)
        devout("Esc-Exit")
        setcolor("GR+/B,GR+/R,,,GR+/B")
        DISPBOX(3,0,21,79)
        KEYBOARD  CHR(0)
        KEYBOARD  CHR(K_RIGHT)
        KEYBOARD  CHR(K_LEFT)

        dbedit(4,1,20,77,afield,'__func_bwX',aPic,aHead,SUBSTR(cBorder,2,1),SUBSTR(cBorder,4,1),
        restscreen(0,0,24,79,scr)
        SETCOLOR(OLDCOLOR)
        if !empty(oldorder)
           dbsetorder(oldorder)
        endif
RETURN

//
procedure  __FUNC_BWX(DE_IDLE,pos_get) // USE BY agBrowse
local scr := savescreen(0,0,24,79),flag:=1,oldcolor:=setcolor("n/bg,n/w,,,n/bg")
local xrow:=row(),xcol:=col(),max_field
Static nRecno
Static cstring:=""
        max_field := len(dbStruct())
        devpos(1,20)
        devoutpict(Recno(),"@zr 999,999","n/bg")
        do case
           case lastkey()==13
                set cursor on
                _x_field := &(FIELD(pos_get))
                do case
                   case  valtype(_x_field)=="D"
                         fgetbcac(row(),col(),_x_field)
                   case  valtype(_x_field)=="N"
                         @ xrow,xcol get _x_field picture aPicture[pos_get]
                         read
                   case  valtype(_x_field)=="C"
                         @ xrow,xcol get _x_field picture aPicture[Pos_get]
                         read
                endcase
                if rlock()
                   replace &(FIELD(pos_get))  WITH _x_field
                   DBCOMMIT()
                endif
                set cursor off
                flag=2
                KEYBOARD CHR(K_RIGHT)
                IF POS_GET >= max_field
                      DBAPPEND()
                      KEYBOARD CHR(K_HOME)
                ENDIF
           case lastkey()==27
                KEYBOARD CHR(0)
                flag = 0
           case lastkey()==K_INS
                   dbappend()
                   KEYBOARD CHR(K_HOME)
           case lastkey()==K_DEL
                if rlock()
                   dbdelete()
                endif
                flag=2
                KEYBOARD CHR(0)

        endcase

       restscreen(0,0,24,79,scr)
       setcolor(oldcolor)
RETURN flag
```

■

ลิสติงที่ 2 โปรแกรมอ่านข้อมูลเพื่อแปลงเป็นเมนู (RDMENU.PRG)

```
/*
       PROGRAM RDMENU.prg
       DESIGN & PROGRAMMING BY AONG
       DATE : 16-08-96
       โปรแกรมอ่านค่าของเมนู
*/

/* DESCRIPTION VAR

   LABEL MENU
   MENUDEF[X]

   DESCRIPTION MENU
   NO.    MENUDESC[X,1]
   COLOR MENUDESC[X,2]
   CALL   MENUDESC[X,3]
   DIR    MENUDESC[X,4]
*/

LOCAL MENUDEF,nMenu := 1.0001,nchoice,NSEQ:=1,PREQLIST[1,3],Pos[5],SCRSAVE[1]
PRIVATE MENUDESC,QLIST[3],PREVMENU[1]

#include "inkey.ch"
#include "color.ch"

DBUSEAREA(,,"MENU")
DBSETINDEX("MENU")
   SETCOLOR("W/N")
   CLEAR
   WHILE(.T.)
     MENUDEF :=  LOAD_MENU(nMenu)

//      nchoice := achoice(7,5,17,60,MENUDEF,.T.,"unc") สามารถเลือก Achoice ทำงานได้
     POS := LOAD_POS(MENUDESC)
     SETCOLOR(POS[5])
     DISPBOX(POS[1],POS[2],POS[3],POS[4])
     DISP_LEVEL(NSEQ,POS)
nchoice := CM_MENU(POS[1]+1,POS[2]+1,POS[3]-1,POS[4]-2,MENUDEF,,@QLIST) ฟังก์ชั่นทำรูปแบบเมนู ใช้แทน Achoice
DO CASE
        CASE LASTKEY()==K_F10
             EXIT
        CASE LASTKEY()==K_ESC
             IF NSEQ > 1
                NSEQ--
                nMenu = PREVMENU[NSEQ]
                QLIST[1]=PREQLIST[NSEQ,1]
                QLIST[2]=PREQLIST[NSEQ,2]
                QLIST[3]=PREQLIST[NSEQ,3]
             ENDIF
             RESTSCREEN(0,0,24,79,SCRSAVE[NSEQ])
        CASE LASTKEY()==K_ENTER
             do case
                case MENUDESC[nchoice,2]=="M"
                     IF NSEQ > LEN(PREVMENU)
                        AADD(PREVMENU,)
                        AADD(PREQLIST,{,,})
                        AADD(SCRSAVE,)
                     ENDIF
                     SCRSAVE[NSEQ]:=SAVESCREEN(0,0,24,79)
                     PREVMENU[NSEQ]=nMenu
                     PREQLIST[NSEQ,1]=QLIST[1]
                     PREQLIST[NSEQ,2]=QLIST[2]
                     PREQLIST[NSEQ,3]=QLIST[3]
                     nMenu=VAL(MENUDESC[nchoice,4])
                     NSEQ++
                     QLIST[1]=NIL
                     QLIST[2]=NIL
                     QLIST[3]=NIL
                 case MENUDESC[nchoice,2]=="P"
             endcase
     ENDCASE
   ENDDO
DBCLOSEALL()
```

→

ลิสติงที่ 2 (ต่อ)

```
SETCOLOR("W/N")
CLEAR
RETURN

// อ่านตำแหน่งบรรทัดและคอลัมน์ ของเมนู
STATIC PROCEDURE LOAD_POS(MENUDESC)
LOCAL NPOS[5],CPOS:=MENUDESC[1,6]
     IF !EMPTY(CPOS)
      NPOS[1]=VAL(SUBSTR(CPOS,1,2))    // TOP
      NPOS[2]=VAL(SUBSTR(CPOS,4,2))    // LEFT
      NPOS[3]=VAL(SUBSTR(CPOS,7,2))    // BOTTOM
      NPOS[4]=VAL(SUBSTR(CPOS,10,2))   // RIGHT
      NPOS[5]=MENUDESC[1,3]            // COLOR
     ELSE
      ALERT("กรุณากำหนดตำแหน่งบรรทัดด้วย")
      NPOS[1]=9
      NPOS[2]=10
      NPOS[3]=17
      NPOS[4]=65
     ENDIF
RETURN NPOS
//  แสดงระดับชั้นของเมนู
STATIC PROCEDURE DISP_LEVEL(NSEQ,POS)
       DEVPOS(POS[1],POS[4]-12)
       DEVOUT("[Level:")
       DEVOUTPICT(NSEQ,"@ZR 99")
       DEVOUT("]")
RETURN
// ดึงข้อมูลของเมนู เก็บไว้ในตัวแปรเพื่อดำเนินการ
STATIC PROCEDURE LOAD_MENU(nMenu)
LOCAL MENUDEF := ARRAY(20),COUNTER
   MENUDESC = ARRAY(20,6)
   // ล้างค่าเมนูก่อนเรียกเมนูอื่นๆ
   FOR COUNTER = 1 TO LEN(MENUDEF)
       MENUDEF[COUNTER]="["+STR(COUNTER,2)+"] "+SPACE(60)
       MENUDESC[COUNTER,1]=0
       MENUDESC[COUNTER,2]=SPACE(01)
       MENUDESC[COUNTER,3]=SPACE(30)
       MENUDESC[COUNTER,4]=SPACE(30)
       MENUDESC[COUNTER,5]=SPACE(30)
       MENUDESC[COUNTER,6]=SPACE(15)
   NEXT COUNTER
   COUNTER = 1
   DBSEEK(nMenu) // หาเมนูเพื่อทำการเก็บเข้าสู่ตัวแปร
   WHILE(LEFT(ALLTRIM(STR(MENUNO)),4) == LEFT(ALLTRIM(STR(nMenu)),4) .AND. !EOF())
      IF COUNTER > LEN(MENUDEF) // กรณีที่มีลำดับของเมนูมากกว่าค่าที่กำหนดก็ให้ทำการสร้าง Array เพิ่มเติมได้
         AADD(MENUDEF,)
         AADD(MENUDESC,{,,,,,})
      ENDIF
      MENUDEF[COUNTER]="["+STR(COUNTER,2)+"] "+MENULABEL
      MENUDESC[COUNTER,1]=MENUNO
      MENUDESC[COUNTER,2]=MENUFLAG
      MENUDESC[COUNTER,3]=MENUCOLOR
      MENUDESC[COUNTER,4]=MENUCALL
      MENUDESC[COUNTER,5]=MENUDIR
      MENUDESC[COUNTER,6]=MENUPOS
      DBSKIP()
      COUNTER++
   ENDDO
RETURN MENUDEF
#include "c_menu.ch"   // ดึงโปรแกรมทำรูปแบบเมนูเข้ามา Compile ด้วย
//  เป็น Procedure ใช้คู่กับ Achoice ถ้าคุณใช้ Achoice ก็ให้พิมพ์ Procedure นี้ลงไปด้วยแต่ถ้าไม่ใช้ ก็ไม่ต้องพิมพ์
procedure unc()
LOCAL FLAG := 2
   do case
      case LASTKEY()==K_ESC
           FLAG = 0
      case lastkey()==K_ENTER
           FLAG = 1
      case lastkey()==K_F10
           FLAG = 0
   endcase
return FLAG
```

ลิสติงที่ 3 โปรแกรมทำหน้าที่ทำรูปแบบเมนู (BROWSE.CH)

```
      program         : c_menu.ch
      Description     : View
      Programming by  : Yukonart chanpanichkijjakarn
      Year            : 1996

*/

//
 static procedure cm_menu(top,left,bottom,right,amenu,flag,alastquit)
 Local  bios_key,mtable_run:=array(8),seq_no:=1,row:=top
 local  scr


      //  กำหนดค่าเริ่มต้นเพื่อเตรียมพร้อมแสดงเมนู
      mtable_run[1]=top
      mtable_run[2]=left
      mtable_run[3]=bottom
      mtable_run[4]=right
      mtable_run[5]=iif(alastquit[1]==nil,seq_no,alastquit[1])
      mtable_run[6]=iif(alastquit[2]==nil,row,alastquit[2])
      mtable_run[7]=iif(alastquit[3]==nil,1,alastquit[3]) // LEFT START
      mtable_run[8]=(right-left)+2

       dispitem("all",amenu,mtable_run) //แสดงรายการทั้งหมด
       dispitem("rev",amenu,mtable_run) //แสดงเฉพาะรายการ

       while empty(flag)                // display only
         dispitem("rev",amenu,mtable_run) //แสดงรายการที่กำหนด
         bios_key=0
         while(bios_key==0) // ตรวจสอบการเลื่อนปุ่มลูกศร
               bios_key     := inkey(1)
         enddo

         // คีย์ที่สามารถหลุดออกจากเมนูได้
         if bios_key==K_ESC .or. bios_key==K_TAB .or. bios_key==K_ALT_S ;
            .or. bios_key==K_F10 .or. bios_key==K_INS .or. bios_key==K_F1  ;
            .or. bios_key==K_F5 .OR. bios_key==K_F3 .or. lastkey()==K_F11 .or.
lastkey()==K_F12

            exit
         endif

         dispitem("nor",amenu,mtable_run)   // แสดงรายการปกติเมื่อเลื่อนเมนูมา 1 ตำแหน่ง

         do case
            case bios_key==K_LEFT
                 if ( (mtable_run[7]-1) > 0 )
                      mtable_run[7]-
                      dispitem("all",amenu,mtable_run)   // กรณีที่กดปุ่มซ้ายเพื่อเลื่อนข้อความ
                 else
                     alert("out of column")
                 endif
            case bios_key==K_RIGHT
                 if ( ((mtable_run[7]+1)+mtable_run[8]) < len(amenu[1]) )
                      mtable_run[7]++
                      dispitem("all",amenu,mtable_run)   // กรณีที่กดปุ่มขวาเพื่อเลื่อนข้อความ
                 else
                       alert("out of column")
                 endif
                 case bios_key==K_PGDN
                       rtk_pgdn(@mtable_run,amenu)
                 case bios_key==K_PGUP
                       rtk_pgup(@mtable_run,amenu)
                 case bios_key==K_UP
                       rtk_up(amenu,@mtable_run)
                 case bios_key==K_DOWN
                       rtk_down(amenu,@mtable_run)
                 case bios_key==K_ENTER .or. bios_key==K_F2 .or. bios_key==K_F5
```

→

ลิสติงที่ 3 (ต่อ)

```
                exit
        endcase


      enddo
alastquit[1] = mtable_run[5]
alastquit[2] = mtable_run[6]
alastquit[3] = mtable_run[7]
dispitem("rev",amenu,mtable_run) //   แสดงรายการกลับสีก่อนออกจากโปรแกรม
return mtable_run[5]

//
static procedure rtk_pgdn(mtable_run,amenu)
local  counter

       if(mtable_run[6]!=mtable_run[3])
          for counter = mtable_run[6] to (mtable_run[3]-1)
              mtable_run[6]++  // row
              mtable_run[5]++  // seq_no
              if len(amenu) < mtable_run[5]
                 mtable_run[6]-=1
                 mtable_run[5]=len(amenu)
                 exit
              endif
          next counter
       else
          for counter = mtable_run[1] to mtable_run[3]
              if(mtable_run[5]+1 <= len(amenu))
                 mtable_run[5]++  // seq_no
              endif
          next counter
       endif
       dispitem("all",amenu,mtable_run) //   แสดงรายการทั้งหมด
return

//
static procedure rtk_pgup(mtable_run,amenu)
local  counter,currow:=mtable_run[6]
       if(mtable_run[6]!=mtable_run[1])
          for counter = mtable_run[1] to (currow-1)
              mtable_run[6]-  // row
              mtable_run[5]-  // seq_no
          next counter
       else
          for counter = mtable_run[1] to (mtable_run[3]+1)
              if((mtable_run[5]-1) >= 1)
                 mtable_run[5]-  // seq_no
              endif
          next counter
       endif
       dispitem("all",amenu,mtable_run) // แสดงรายการทั้งหมด
return


//
static procedure rtk_up(amenu,table_run)   // ปุ่มเลื่อนลูกศรขึ้น


         if table_run[5] <= 1 // ตรวจสอบว่าไปถึงรายการแรกหรือเปล่า
            table_run[5]=1
         else
            table_run[6]- // row
            table_run[5]- // seq_no
         endif

         if table_run[6] < table_run[1] // ตรวจสอบถ้าเลื่อนรายการจนหมดที่แสดงได้ในหนึ่งหน้าจอ
            table_run[6]=table_run[1]
            dispitem("all",amenu,table_run) //   แสดงรายการทั้งหมด
         endif
```

→

ลิสติงที่ 3 (ต่อ)

```
return
//
static procedure rtk_down(amenu,table_run)   // ปุ่มเลื่อนลูกศรลง

         if table_run[5] >=  len(amenu)
            table_run[5] := len(amenu)
         else
            table_run[6]++ // row
            table_run[5]++ // seq_no
         endif

         if table_run[6] > table_run[3] //
            table_run[6] :=table_run[3]
            dispitem("all",amenu,table_run) //  แสดงข้อมูลรายการทั้งหมด
         endif

return

//
static procedure dispitem(process_type,amenu,mtable_run)
local counter,nlen:=len(amenu),xmenu:=mtable_run[1]-(mtable_run[6]-mtable_run[5]),oldcolor :=
setcolor()
      if xmenu<=0
         xmenu = 1
      endif
      do case
         case process_type == "all"   //   แสดงรายการทั้งหมด
              for counter =  mtable_run[1] to mtable_run[3]
                  devpos(counter,mtable_run[2])
                  devout(repl(" ",mtable_run[4]-mtable_run[2]))
              next counter

              for counter = mtable_run[1] to mtable_run[3]
                  if xmenu <= nlen
                    devpos(counter,mtable_run[2])
                    devout(substr(amenu[xmenu],mtable_run[7],mtable_run[8]))
                  endif
                  xmenu++
              next counter
         case process_type == "rev"    //  แสดงรายการแบบกลับสี
              COLORSELECT(CLR_ENHANCED)
                devpos(mtable_run[6],mtable_run[2])
                devout(substr(amenu[mtable_run[5]],mtable_run[7],mtable_run[8]))
              COLORSELECT(CLR_STANDARD)
         other                              //    แสดงรายการแบบปกติ
                devpos(mtable_run[6],mtable_run[2])
                devout(substr(amenu[mtable_run[5]],mtable_run[7],mtable_run[8]))
      endcase
return
```